

BBBBBBBBBBBB		AAAAAAAAAA		DDDDDDDDDD		BBBBBBBBBBBB		LLL		KKK		KKK
BBBBBBBBBBBB		AAAAAAAAAA		DDDDDDDDDD		BBBBBBBBBBBB		LLL		KKK		KKK
BBBBBBBBBBBB		AAAAAAAAAA		DDDDDDDDDD		BBBBBBBBBBBB		LLL		KKK		KKK
BBB	BBB	AAA	AAA	DDD	DDD	BBB	BBB	LLL		KKK		KKK
BBB	BBB	AAA	AAA	DDD	DDD	BBB	BBB	LLL		KKK		KKK
BBB	BBB	AAA	AAA	DDD	DDD	BBB	BBB	LLL		KKK		KKK
BBB	BBB	AAA	AAA	DDD	DDD	BBB	BBB	LLL		KKK	KKK	KKK
BBB	BBB	AAA	AAA	DDD	DDD	BBB	BBB	LLL		KKK	KKK	KKK
BBB	BBB	AAA	AAA	DDD	DDD	BBB	BBB	LLL		KKK	KKK	KKK
BBBBBBBBBBBB		AAA	AAA	DDD	DDD	BBBBBBBBBBBB		LLL		KKK	KKK	KKK
BBBBBBBBBBBB		AAA	AAA	DDD	DDD	BBBBBBBBBBBB		LLL		KKKKKKKK		
BBBBBBBBBBBB		AAA	AAA	DDD	DDD	BBBBBBBBBBBB		LLL		KKKKKKKK		
BBB	BBB	AAAAAAAAAAAA		DDD	DDD	BBB	BBB	LLL		KKK	KKK	KKK
BBB	BBB	AAAAAAAAAAAA		DDD	DDD	BBB	BBB	LLL		KKK	KKK	KKK
BBB	BBB	AAAAAAAAAAAA		DDD	DDD	BBB	BBB	LLL		KKK	KKK	KKK
BBB	BBB	AAA	AAA	DDD	DDD	BBB	BBB	LLL		KKK	KKK	KKK
BBB	BBB	AAA	AAA	DDD	DDD	BBB	BBB	LLL		KKK	KKK	KKK
BBB	BBB	AAA	AAA	DDD	DDD	BBB	BBB	LLL		KKK	KKK	KKK
BBB	BBB	AAA	AAA	DDD	DDD	BBB	BBB	LLL		KKK	KKK	KKK
BBBBBBBBBBBB		AAA	AAA	DDDDDDDDDD		BBBBBBBBBBBB		LLLLLLLLLLLL		KKK	KKK	KKK
BBBBBBBBBBBB		AAA	AAA	DDDDDDDDDD		BBBBBBBBBBBB		LLLLLLLLLLLL		KKK	KKK	KKK
BBBBBBBBBBBB		AAA	AAA	DDDDDDDDDD		BBBBBBBBBBBB		LLLLLLLLLLLL		KKK	KKK	KKK

```
SSSSSSSS  CCCCCCCC  AAAAAA  NN  NN  FFFFFFFF  IIIIII  LL  EEEEEEEEE
SSSSSSSS  CCCCCCCC  AAAAAA  NN  NN  FFFFFFFF  IIIIII  LL  EEEEEEEEE
SS  SS  CC  CC  AA  AA  NN  NN  FF  II  LL  EE
SS  SS  CC  CC  AA  AA  NN  NN  FF  II  LL  EE
SS  SS  CC  CC  AA  AA  NN  NN  FF  II  LL  EE
SSSSSS  CC  CC  AA  AA  NN  NN  FF  II  LL  EEEEEEE
SSSSSS  CC  CC  AA  AA  NN  NN  FF  II  LL  EEEEEEE
SS  CC  CC  AA  AA  NN  NN  FF  II  LL  EE
SS  CC  CC  AA  AA  NN  NN  FF  II  LL  EE
SSSSSS  CCCCCCCC  AA  AA  NN  NN  FF  IIIIII  LL  EEEEEEEEE
SSSSSS  CCCCCCCC  AA  AA  NN  NN  FF  IIIIII  LL  EEEEEEEEE
LL  IIIIII  SSSSSSS
LL  IIIIII  SSSSSSS
LL  II  SS
LL  II  SS
LL  II  SS
LL  II  SSSSSS
LL  II  SSSSSS
LL  II  SS
LL  II  SS
LL  II  SS
LL  II  SS
LL  IIIIII  SSSSSSS
LLLLLLLLLL  IIIIII  SSSSSSS
LLLLLLLLLL  IIIIII  SSSSSSS
```

```
1 0001 0 MODULE SCANFILE (  
2 0002 0 IDENT = 'V04-000'  
3 0003 0 ) =  
4 0004 1 BEGIN  
5 0005 1  
6 0006 1  
7 0007 1 *****  
8 0008 1 *  
9 0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *  
10 0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *  
11 0011 1 * ALL RIGHTS RESERVED. *  
12 0012 1 *  
13 0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *  
14 0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *  
15 0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *  
16 0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *  
17 0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *  
18 0018 1 * TRANSFERRED. *  
19 0019 1 *  
20 0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *  
21 0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *  
22 0022 1 * CORPORATION. *  
23 0023 1 *  
24 0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *  
25 0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *  
26 0026 1 *  
27 0027 1 *****  
28 0028 1  
29 0029 1  
30 0030 1 ++  
31 0031 1 FACILITY:  
32 0032 1 DYNAMIC BAD BLOCK UTILITY  
33 0033 1 ABSTRACT:  
34 0034 1 THIS PROCESS EXAMINES FILES SUSPECTED OF CONTAINING BAD  
35 0035 1 DISK BLOCKS. THOSE DISK BLOCKS VERIFIED TO BE BAD ARE ADDED  
36 0036 1 TO THE BAD BLOCK FILE. THE OTHERS ARE RETURNED TO THE VOLUME  
37 0037 1 FOR REUSE.  
38 0038 1 ENVIRONMENT:  
39 0039 1 VAX/VMS OPERATING SYSTEM, VERSION 1.0  
40 0040 1 AUTHOR:THOMAS G. DOPIRAK , CREATION DATE:5/16/78  
41 0041 1  
42 0042 1 MODIFIED BY:  
43 0043 1  
44 0044 1 V0002 ACG0059 Andrew C. Goldstein, 21-Aug-1979 20:45  
45 0045 1 Fix repeated write/read test so it really repeats  
46 0046 1  
47 0047 1 --
```



```
49 0048 1 |
50 0049 1 | TABLE OF CONTENTS:
51 0050 1 |
52 0051 1 |
53 0052 1 | FORWARD ROUTINE
54 0053 1 |     SCAN:NOVALUE,          !MAIN PROGRAM OF FILE PROCESSING LOGIC
55 0054 1 |     GROUP_BLOCKTEST,      !ROUTINE TESTS 'GROUPS' OF BLOCKS
56 0055 1 |     GROUP_RETURN,         !EXAMINES A GROUP OF BLOCKS WHEN ERROR DETECTED
57 0056 1 |     CHECK_BADSTATUS,      !DETERMINES WHETHER A STATUS INDICATES A BAD BLOCK
58 0057 1 |     NORMAL_COMPLETE:NOVALUE, !CALLED AFTER NORMAL PROCESSING OF A FILE
59 0058 1 |     ERROR_COMPLETE:NOVALUE, !CALLED AFTER ABNORMAL PROCESSING OF A FILE
60 0059 1 |     TRUNCATE,             !TRUNCATES A FILE CONTAINING NO BAD BLOCKS
61 0060 1 |     TRUNCATE_BAD,         !TRUNCATES A FILE CONTAINING A BAD BLOCK
62 0061 1 |     BLOCKTEST,           !TESTS AN INDIVIDUAL BLOCK
63 0062 1 |     POSITION_TO_EOF,       !ACCESSES A FILE AND DISCOVERS ITS SIZE
64 0063 1 |     DO_QIOW,              !DOES QIOW'S ANS CHECKS STATUS
65 0064 1 |     ;
66 0065 1 |
67 0066 1 |
68 0067 1 | INCLUDE FILES:
69 0068 1 |
70 0069 1 |
71 0070 1 |     LIBRARY 'SYS$LIBRARY:LIB.L32';
72 0071 1 |
73 0072 1 | MACROS:
74 0073 1 |
75 0074 1 |
76 0075 1 | MACRO
77 0076 1 |     DIRECTORY_ID=FIB[FIB$W_DID]%,          !START OF DIRECTORY ID
78 M 0077 1 |     CSTRING[] = (UPLIT_BYTE(%CHARCOUNT(%STRING(%REMAINING))),
79 0078 1 |     %STRING(%REMAINING)) )%
80 0079 1 |     ;
81 0080 1 |
82 0081 1 | VARIOUS DEFINITIONS
83 0082 1 |
84 0083 1 |
85 0084 1 | LITERAL
86 0085 1 |     BLOCK_TEST_SIZE=15,          !NUMBER OF BLOCKS IN A GROUP
87 0086 1 |     GROUP_SIZE=BLOCK_TEST_SIZE*512, !NUMBER OF BYTES IN A GROUP
88 0087 1 |     TRIALS_TO_SUC=3;             !NUMBER OF TIMES A BLOCK MUST BE SUCCESSFULLY BE WRITTEN
89 0088 1 |                                     !BEFORE ITS DECLARED NO BAD
90 0089 1 |
91 0090 1 | OWN STORAGE:
92 0091 1 |
93 0092 1 |
94 0093 1 | OWN
95 0094 1 |
96 0095 1 |     !BLOCKS OF TEST DATA
97 0096 1 |
98 0097 1 |     DISK_TEXT:VECTOR[GROUP_SIZE,BYTE],
99 0098 1 |     GROUP_TEST_DATA:VECTOR[GROUP_SIZE,BYTE],
100 0099 1 |
101 0100 1 |     READ_FAIL,                !LOGICAL INDICATING IF GROUP FAILED ON READ
102 0101 1 |     TRUNC_BLOCK,              !FIRST VBN ACTUALLY REMOVED IN TRUNCATE
103 0102 1 |     BAD_COUNT:INITIAL(0),     !COUNT OF BAD BLOCKS FOUND
104 0103 1 |     STARTING_BLOCK,           !FIRST VBN IN GROUP
105 0104 1 |     LAST_BLOCK,               !LAST BLOCK IN A GROUP
```

```

: 106      0105 1      IOSB:VECTOR[4,WORD] INITIAL(0,0),      !IOSB FOR DISK OPERATIONS
: 107      0106 1      FIB:BLOCK[FIB$C_LENGTH,BYTE]          !FILE IDENTIFICATION BLOCK
: 108      0107 1      ;
: 109      0108 1      ;
: 110      0109 1      !EQUATED SYMBOLS:
: 111      0110 1      ;
: 112      0111 1      ;
: 113      0112 1      BIND
: 114      0113 1      ;
: 115      0114 1      ;
: 116      0115 1      !SYMBOLS FOR TYPES OF BLOCK TEST RESULTS
: 117      0116 1      ;
: 118      0117 1      NORMAL_STS=0,      !TEST COMPLETED NORMALLY
: 119      0118 1      ERROR_STS=1,      !NON-RECOVERABLE ERROR
: 120      0119 1      BAD_STS=2,      !BAD BLOCK INDICATED
: 121      0120 1      TRUE=1,
: 122      0121 1      FALSE=0,
: 123      0122 1      FIB_DESC=UPLIT(FIB$C_LENGTH,FIB)
: 124      0123 1      ;
: 125      0124 1      ;
: 126      0125 1      !EXTERNAL REFERENCES:
: 127      0126 1      ;
: 128      0127 1      ;
: 129      0128 1      EXTERNAL
: 130      0129 1      CHANNEL:WORD,
: 131      0130 1      MBX_CHANNEL:WORD,      !CHANNEL TO F11ACP MAILBOX
: 132      0131 1      ACP_MAIL:BLOCK[,BYTE],      !BUFFER FROM F11ACP
: 133      0132 1      OLD_UCB
: 134      0133 1      ;
: 135      0134 1      ;
: 136      0135 1      EXTERNAL ROUTINE
: 137      0136 1      SET_UCB;

```

```
: 139 0137 1 GLOBAL ROUTINE SCAN:NOVALUE=
: 140 0138 1
: 141 0139 1 !++
: 142 0140 1 FUNCTIONAL DESCRIPTION:
: 143 0141 1
: 144 0142 1 MAIN ROUTINE FOR FILE PROCESSING. CONTROLS THE
: 145 0143 1 EXAMINATION OF THE FILE IN GROUPS AND THE RETURN
: 146 0144 1 OF THE FILES BLOCKS.
: 147 0145 1
: 148 0146 1 FORMAL PARAMETERS:
: 149 0147 1
: 150 0148 1 NONE
: 151 0149 1
: 152 0150 1 IMPLICIT INPUTS:
: 153 0151 1
: 154 0152 1 CHANNEL: CHANNEL TO SUSPECT DEVICE
: 155 0153 1 ACP_MAIL: MAIL FROM F11ACP
: 156 0154 1
: 157 0155 1 IMPLICIT OUTPUTS:
: 158 0156 1
: 159 0157 1 NONE
: 160 0158 1
: 161 0159 1 ROUTINE VALUE:
: 162 0160 1 COMPLETION CODES:
: 163 0161 1
: 164 0162 1 NONE
: 165 0163 1
: 166 0164 1 SIDE EFFECTS:
: 167 0165 1
: 168 0166 1 THE SUSPECT FILE IS RETURNED TO THE SYSTEM, BLOCKWISE.
: 169 0167 1
: 170 0168 1 --
: 171 0169 1
: 172 0170 2 BEGIN
: 173 0171 2
: 174 0172 2 !++
: 175 0173 2 CLEAR THE FIB
: 176 0174 2 --
: 177 0175 2
: 178 0176 2 CH$FILL(0,FIB$C_LENGTH,FIB);
: 179 0177 2
: 180 0178 2 !*
: 181 0179 2 !INITIALIZE ACCESS TO A FILE AND INITIALIZE LAST_BLOCK
: 182 0180 2
: 183 0181 2 IF
: 184 0182 2 NOT POSITION_TO_EOF()
: 185 0183 2 THEN
: 186 0184 2 RETURN;
: 187 0185 2
: 188 0186 2 !*
: 189 0187 2 !LOOP THROUGH ALL GROUPS IN THE FILE
: 190 0188 2
: 191 0189 2 WHILE TRUE DO
: 192 0190 2 BEGIN
: 193 0191 2
: 194 0192 2 !*
: 195 0193 2 !FIND START OF GROUP TO TEST
```



```

196 0194
197 0195
198 0196
199 0197
200 0198
201 0199
202 0200
203 0201
204 0202
205 0203
206 0204
207 0205
208 0206
209 0207
210 0208
211 0209
212 0210
213 0211
214 0212
215 0213
216 0214
217 0215
218 0216
219 0217
220 0218
221 0219
222 0220
223 0221
224 0222
225 0223
226 0224
227 0225
228 0226
229 0227
230 0228
231 0229
232 0230
233 0231
234 0232
235 0233
236 0234
237 0235
238 0236
239 0237
240 0238
241 0239
242 0240
243 0241
244 0242
245 0243
246 0244
247 0245
248 0246
249 0247
250 0248
251 0249

IF .LAST_BLOCK LSSU BLOCK_TEST_SIZE
THEN STARTING_BLOCK=1
ELSE STARTING_BLOCK=.LAST_BLOCK-BLOCK_TEST_SIZE+1;

! *
! TEST GROUP OF BLOCKS
! ACTION DEPENDS UPON WHETHER ANY 'BAD' BLOCKS FOUND
CASE CHECK_BADSTATUS(GROUP_BLOCKTEST()) FROM NORMAL_STS TO BAD_STS OF
SET
[NORMAL_STS]: !SUCCESSFUL TEST
++
CHECK TO SEE IF FINISHED
IF --
THEN .STARTING_BLOCK EQLU 1
THEN
BEGIN
NORMAL_COMPLETE();
RETURN
END
ELSE
LAST_BLOCK=.STARTING_BLOCK-1;
[ERROR_STS]: !ERROR BUT NOT BADBLOCK ERROR
BEGIN
ERROR_COMPLETE();
RETURN
END;
[BAD_STS]: !BADBLOCK FOUND,SCAN INDIVIDUAL BLOCKS
BEGIN
IF
NOT GROUP_RETURN()
THEN
BEGIN
ERROR_COMPLETE();
RETURN
END;
IF
(.STARTING_BLOCK EQLU 1) OR
(.TRUNC_BLOCK LEQ 1)
THEN
BEGIN
NORMAL_COMPLETE();
RETURN
END
ELSE
LAST_BLOCK=.TRUNC_BLOCK-1
END;
TES;
END
```

```
.TITLE SCANFILE
.IDENT \V04-000\

.PSECT $SPLITS,NOWRT,NOEXE,2

00000040 00000 P.AAA: .LONG 64
00000000 00004 .ADDRESS FIB

.PSECT $OWNS,NOEXE,2

00000 DISK_TEXT:
01E00 GROUP_TEST_DATA:
03C00 READ_FAIL:
03C04 TRUNC_BLOCK:
00000000 03C08 BAD_COUNT:
03C0C STARTING_BLOCK:
03C10 LAST_BLOCK:
00000000 00000000 03C14 IOSB:
03C1C FIB:

NORMAL_STS= 0
ERROR_STS= 1
BAD_STS= 2
TRUE= 1
FALSE= 0
FIB_DESC= P.AAA

.EXTRN CHANNEL, MBX_CHANNEL
.EXTRN ACP_MAIL, OLD_UCB
.EXTRN SET_UCB

.PSECT $CODES,NOWRT,2

.ENTRY SCAN, Save R2,R3,R4,R5,R6
MOVAB STARTING_BLOCK, R6
MOVCS #0, (SP), #0, #64, FIB

CALLS #0, POSITION_TO_EOF
BLBC R0, 11$
CMPL LAST_BLOCK, #15
BGEQU 2$
MOVL #1, STARTING_BLOCK
BRB 3$
SUBL3 #14, LAST_BLOCK, STARTING_BLOCK
CALLS #0, GROUP_BLOCKTEST
PUSHL R0
CALLS #1, CHECK_BADSTATUS
CASEL R0, #0, #2
.WORD 5$-4$,-
      7$-4$,-

0040 8F 00 56 0000' 007C 00000 .ENTRY SCAN, Save R2,R3,R4,R5,R6
      6E 00 9E 00002 MOVAB STARTING_BLOCK, R6
      10 00 2C 00007 MOVCS #0, (SP), #0, #64, FIB
      0000V CF 00 A6 0000E
      59 00 FB 00010 CALLS #0, POSITION_TO_EOF
      OF 04 A6 D1 00018 1$: BLBC R0, 11$
      66 05 1E 0001C CMPL LAST_BLOCK, #15
      04 01 D0 0001E BGEQU 2$
      66 05 11 00021 MOVL #1, STARTING_BLOCK
      04 A6 0E C3 00023 2$: BRB 3$
      0000V CF 00 FB 00028 3$: SUBL3 #14, LAST_BLOCK, STARTING_BLOCK
      0000V CF 50 DD 0002D CALLS #0, GROUP_BLOCKTEST
      02 01 FB 0002F PUSHL R0
      0012 50 CF 00034 CALLS #1, CHECK_BADSTATUS
      001A 0006 00038 4$: CASEL R0, #0, #2
      .WORD 5$-4$,-
            7$-4$,-
```


SCANFILE
V04-000

K 15
15-Sep-1984 23:36:57
14-Sep-1984 11:54:33

VAX-11 Bliss-32 V4.0-742
[BADBLK.SRC]SCANFILE.B32;1

Page 7
(3)

		01		66	D1	0003E	5\$:	CMPL	6\$-4\$		
				20	13	00C41		BEQL	STARTING_BLOCK, #1	...	0213
04	A6	66		01	C3	00043		SUBL3	9\$...	
				CE	11	00048		BRB	#1, STARTING_BLOCK, LAST_BLOCK	...	0220
	0000V	CF		00	FB	0004A	6\$:	CALLS	1\$...	0212
		06		50	E8	0004F		BLBS	#0, GROUP_RETURN	...	0230
	0000V	CF		00	FB	00052	7\$:	CALLS	R0, 8\$...	
					04	00057		RET	#0, ERROR_COMPLETE	...	0233
		01		66	D1	00058	8\$:	CMPL	STARTING_BLOCK, #1	...	0232
				06	13	0005B		BEQL	9\$...	0237
		01	F8	A6	D1	0005D		CMPL	TRUNC_BLOCK, #1	...	0238
				06	14	00061		BGTR	10\$...	
	0000V	CF		00	FB	00063	9\$:	CALLS	#0, NORMAL_COMPLETE	...	0241
					04	00068		RET		...	0240
04	A6	F8	A6	01	C3	00069	10\$:	SUBL3	#1, TRUNC_BLOCK, LAST_BLOCK	...	0245
				A7	11	0006F		BRB	1\$...	0189
				04	00071	11\$:		RET		...	0249

; Routine Size: 114 bytes, Routine Base: \$CODE\$ + 0000

```
254 0250 1 ROUTINE POSITION_TO_EOF=
255 0251 1
256 0252 1 !++
257 0253 1 FUNCTIONAL DESCRIPTION:
258 0254 1
259 0255 1 ROUTINE INITIALIZES THE FIB, ACCESSES THE FILE WHOSE
260 0256 1 FID IS THE ACP_MAIL, AND DETERMINES THE FILES LENGTH
261 0257 1 IN BLOCKS
262 0258 1
263 0259 1 FORMAL PARAMETERS:
264 0260 1
265 0261 1 NONE
266 0262 1
267 0263 1 IMPLICIT INPUTS:
268 0264 1
269 0265 1 ACP_MAIL[BBSSW_FID]: FILE ID OF SUSPECT FILE
270 0266 1
271 0267 1 IMPLICIT OUTPUTS:
272 0268 1
273 0269 1 LAST_BLOCK: TOTAL NUMBER OF BLOCKS IN FILE
274 0270 1 FIB: ASSORTED FIELDS SET BY IOS_ACCESS
275 0271 1
276 0272 1 ROUTINE VALUE:
277 0273 1 COMPLETION CODES:
278 0274 1
279 0275 1 IF IOS_ACCESS FAILS THEN THAT CODE IS RETURNED
280 0276 1
281 0277 1 SIDE EFFECTS:
282 0278 1
283 0279 1 NONE
284 0280 1
285 0281 1 --
286 0282 1
287 0283 2 BEGIN
288 0284 2
289 0285 2 OWN
290 0286 2
291 0287 2 STAT_BLOCK:VECTOR[5,WORD], !SPACE FOR FILE STATISTICS BLOCK
292 0288 2 !RETURNED BY IOS_ACCESS
293 0289 2 ATTRIBUTES:VECTOR[3]
294 0290 2 INITIAL(ATRSC_STATBLK*16+10,STAT_BLOCK,0)
295 0291 2 ;
296 0292 2
297 0293 2 !*
298 0294 2 !SET FILE ACCESS ATTRIBUTES
299 0295 2
300 0296 2 FIB[FIBSV_WRITE]=1;
301 0297 2 FIB[FIBSV_TRUNC]=1;
302 0298 2 !*
303 0299 2 !PUSH FILE ID INTO FIB
304 0300 2
305 0301 2 CH$MOVE(6,ACP_MAIL[BBSSW_FID],FIB[FIBSW_FID]);
306 0302 2
307 0303 2 !*
308 0304 2 !OPEN THE SPECIFIED FILE AND GET ITS SIZE IN BLOCKS
309 0305 2
310 0306 2 DO_QIOW(IOS_ACCESS+IOSM_ACCESS,FIB_DESC,0,0,0,ATTRIBUTES);
```

; Routine Size: 55 bytes. Routine Base: \$CODE\$ + 0072


```

321 0316 1 ROUTINE TRUNCATE(VBN)=
322 0317 1
323 0318 1 !*
324 0319 1 FUNCTIONAL DESCRIPTION:
325 0320 1
326 0321 1 ROUTINE TRUNCATES OF THE END OF THE CURRENT FILE
327 0322 1 STARTING AT THE INDICATED BLOCK NUMBER. BECAUSE OF
328 0323 1 CLUSTERING NOT ALL BLOCKS REQUESTED MAY BE TRUNCATED
329 0324 1 .LAST BLOCK TRUNCATED IS PLACED INTO TRUNC_BLOCK.
330 0325 1
331 0326 1 FORMAL PARAMETERS:
332 0327 1
333 0328 1 VBN: VIRTUAL BLOCK AT WHICH TO START TRUNCATE
334 0329 1
335 0330 1 IMPLICIT INPUTS:
336 0331 1
337 0332 1 NONE
338 0333 1
339 0334 1 IMPLICIT OUTPUTS:
340 0335 1
341 0336 1 NONE
342 0337 1
343 0338 1 ROUTINE VALUE:
344 0339 1 COMPLETION CODES:
345 0340 1
346 0341 1 STATUS OF IOS_MODIFY OPERATION IS RETURNED
347 0342 1
348 0343 1 SIDE EFFECTS:
349 0344 1
350 0345 1 NONE
351 0346 1
352 0347 1 --
353 0348 1
354 0349 2 BEGIN
355 0350 2 LOCAL
356 0351 2 STATUS;
357 0352 2
358 0353 2 !*
359 0354 2 !SET BLOCK TO TRUNCATE AT
360 0355 2
361 0356 2 FIB[FIB$L_EXVBN]=.VBN;
362 0357 2
363 0358 2 !*
364 0359 2 !TRUNCATE A PIECE OFF OF FILE
365 0360 2
366 0361 2 STATUS=DO_QIOW(IOS_MODIFY,FIB_DESC,0,0,0,0);
367 0362 2 !*
368 0363 2 !CLEAR SIZE FIELD
369 0364 2
370 0365 2 FIB[FIB$L_EXSZ]=0;
371 0366 2
372 0367 2 !*
373 0368 2 !CHECK FOR ROUNDING FROM CLUSTERING
374 0369 2
375 0370 2 IF
376 0371 2 .VBN NEQ .FIB[FIB$L_EXVBN]
377 0372 2 THEN

```

```

: 378      0373 2
: 379      0374 2
: 380      0375 2
: 381      0376 2
: 382      0377 2
: 383      0378 2
: 384      0379 1 END;

      TRUNC_BLOCK=.FIB[FIB$L_EXVBN]
ELSE
      TRUNC_BLOCK=.VBN;
      RETURN .STATUS

```

```

                                0004 0000 TRUNCATE:
                                .WORD  Save R2
                                MOVAB  FIB+28, R2
                                MOVL   VBN, FIB+28
                                CLRQ   -(SP)
                                CLRQ   -(SP)
                                PUSHAB FIB_DESC
                                PUSHL  #54
                                CALLS  #6, DO_Q10W
                                CLRL   FIB+24
                                CMPL   VBN, FIB+28
                                BEQL   1$
                                MOVL   FIB+28, TRUNC_BLOCK
                                RET
                                MOVL   VBN, TRUNC_BLOCK
                                RET

```

```

: 0316
: 0356
: 0361
:
: 0365
: 0371
: 0373
: 0375
: 0379

```

; Routine Size: 46 bytes, Routine Base: \$CODE\$ + 00A9

```
386 0380 1 ROUTINE TRUNCATE_BAD(VBN)=
387 0381 1
388 0382 1 ++
389 0383 1 FUNCTIONAL DESCRIPTION:
390 0384 1
391 0385 1 TRUNCATE BAD PERFORMS 2 TRUNCATIO OPERATIONS.
392 0386 1 ALL BLOCKS AFTER(HIGHER VBN'S) ARE RETURNED TO
393 0387 1 THE SYSTEM VIA A CALL TO TRUNCATE. THE CURRENT
394 0388 1 VBN KNOWN AS 'BAD' IS TRUNCATED OFF THE CURRENT FILE
395 0389 1 AND ONTO THE BAD BLOCK FILE. DUE TO CLUSTERING, MORE BLOCKS
396 0390 1 THAN REQUESTED MAY BE ADDED TO THE BAD BLOCK FILE AND
397 0391 1 TRUNC_BLOCK IS SET TO THE LAST BLOCK ADDED.
398 0392 1
399 0393 1 FORMAL PARAMETERS:
400 0394 1
401 0395 1 VBN: VIRTUAL BLOCK NUMBER OF BLOCK TO MARK BAD
402 0396 1
403 0397 1 IMPLICIT INPUTS:
404 0398 1
405 0399 1 NONE
406 0400 1
407 0401 1 IMPLICIT OUTPUTS:
408 0402 1
409 0403 1 TRUNC_BLOCK: LAST BLOCK(LOWEST VBN) ADDED TO BAD BLOCK FILE
410 0404 1
411 0405 1 ROUTINE VALUE:
412 0406 1 COMPLETION CODES:
413 0407 1
414 0408 1 IF EITHER TRUNCATE OPERATION FAILS THEN THAT STATUS IF RETURNED
415 0409 1
416 0410 1 SIDE EFFECTS:
417 0411 1
418 0412 1 NONE
419 0413 1
420 0414 1 --
421 0415 1
422 0416 2 BEGIN
423 0417 2 LOCAL
424 0418 2 STATUS;
425 0419 2
426 0420 2 BAD_COUNT=.BAD_COUNT+1;
427 0421 2
428 0422 2 ! TRUNCATE OFF GOOD PORTIONS OF FILE
429 0423 2
430 0424 2 STATUS=TRUNCATE(.VBN+1);
431 0425 2 IF
432 0426 2 (.STATUS NEQ SS$NORMAL) AND
433 0427 3 (.STATUS NEQ SS$ENDOFFILE)
434 0428 2 THEN
435 0429 2 RETURN .STATUS;
436 0430 2
437 0431 2 ! SET BLOCK TO TRUNCATE AT
438 0432 2
439 0433 2 FIB[FIB$L_EXVBN]=.VBN;
440 0434 2
441 0435 2
442 0436 2 ! NOTE RETURN IS TO BAD BLOCK FILE
```



```
443 0437 2
444 0438 FIB[FIB$V_MARKBAD]=1;
445 0439
446 0440 !*
447 0441 ! TRUNCATE A PIECE OFF OF FILL
448 0442
449 0443 STATUS=DO_QIOW(IO$_MODIFY,FIB_DESC,0,0,0,0);
450 0444 !*
451 0445 ! CLEAR SIZE FIELD
452 0446
453 0447 FIB[FIB$L_EXSZ]=0;
454 0448
455 0449 !*
456 0450 ! CHECK FOR ROUNDING FROM CLUSTERING
457 0451
458 0452 IF
459 0453 .VBN NEQ .FIB[FIB$L_EXVBN]
460 0454 THEN
461 0455 TRUNC_BLOCK=.FIB[FIB$L_EXVBN]
462 0456 ELSE
463 0457 TRUNC_BLOCK=.VBN;
464 0458
465 0459
466 0460 !*
467 0461 ! CLEAR MARK BAD INDICATOR
468 0462
469 0463 FIB[FIB$V_MARKBAD]=0;
470 0464 RETURN .STATUS
471 0465
472 0466 1 END;
```

```
0004 0000 TRUNCATE_BAD:
WORD Save R2
MOVAB FIB+28, R2
INCL BAD_COUNT
ADDL3 #1, VBN, -(SP)
CALLS #1, TRUNCATE
CMPL STATUS, #1
BEQL 1$
CMPL STATUS, #2160
BNEQ 4$
MOVL VBN, FIB+28
BISB2 #4, FIB+23
CLRQ -(SP)
CLRQ -(SP)
PUSHAB FIB_DESC
PUSHL #54
CALLS #6, DO_QIOW
CLRL FIB+24
CMPL VBN, FIB+28
BEQL 2$
MOVL FIB+28, TRUNC_BLOCK
BRB 3$

7E 04 AC 0000' CF 9E 00002
BF AF 01 A2 D6 00007
01 FB 0000F 01 C1 0000A
50 D1 00013 01 FB 0000F
09 13 00016 50 D1 00013
50 D1 00018 2F 12 0001F
04 AC D0 00021 1$: 04 88 00025
FB A2 04 88 00025
7E 7C 00029 7E 7C 0002B
0000' CF 9F 0002D
0000V CF 06 DD 00031
62 FC 06 FB 00033
04 AC D4 00038
AC D1 0003B
06 13 0003F
CC A2 62 D0 00041
05 11 00045
```

```
Save R2
FIB+28, R2
BAD_COUNT
#1, VBN, -(SP)
#1, TRUNCATE
STATUS, #1
1$
STATUS, #2160
4$
VBN, FIB+28
#4, FIB+23
-(SP)
-(SP)
FIB_DESC
#54
#6, DO_QIOW
FIB+24
VBN, FIB+28
2$
FIB+28, TRUNC_BLOCK
3$
```

```
0380
0420
0424
0426
0427
0433
0438
0443
0447
0453
0455
```

SCANFILE
V04-000

E 16
15-Sep-1984 23:36:57 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 11:54:33 [BADBLK.SRC]SCANFILE.B32;1

Page 14
(7)

CC	A2	04	AC	DO	00047	2\$:	MOVL	VBN, TRUNC_BLOCK
FB	A2		04	8A	0004C	3\$:	BICB2	#4, FIB+23
				04	00050	4\$:	RET	

: 0457
: 0463
: 0466

; Routine Size: 81 bytes, Routine Base: \$CODE\$ + 00D7

```
0467 1 ROUTINE BLOCKTEST(VBN)=
0468 1
0469 1 **
0470 1 FUNCTIONAL DESCRIPTION:
0471 1
0472 1     THIS ROUTINE TESTS A SINGLE VIRTUAL BLOCK
0473 1     FOR 'BADNESS'. THE ROUTINE READS THE BLOCK A NUMBER
0474 1     OF TIMES, CHECKING FOR A DATA SENSITIVE CONDITION, AND THEN
0475 1     WRITES AND READS BACK THE WORST CASE PATTERN. UPON ANY
0476 1     ABNORMAL CONDITION THE ROUTINE EXITS WITH THAT STATUS.
0477 1
0478 1 FORMAL PARAMETERS:
0479 1
0480 1     VBN: VIRTUAL BLOCK TO BE TESTED
0481 1
0482 1 IMPLICIT INPUTS:
0483 1
0484 1     READ_FAIL: A LOGICAL VARIABLE, WHEN TRUE INDICATES THAT
0485 1     GROUP BLOCKTEST ENCOUNTERED AN ERROR WHILE READING THE
0486 1     USER DATA ON THE CURRENT GROUP. THIS DIRECTS BLOCKTEST
0487 1     TO READ THE INDIVIDUAL BLOCKS BEFORE OVERWRITING THEM
0488 1
0489 1 IMPLICIT OUTPUTS:
0490 1
0491 1     NONE
0492 1
0493 1 ROUTINE VALUE:
0494 1 COMPLETION CODES:
0495 1
0496 1     IF ANY QIOW FAILS THEN ITS STATUS IS RETURNED
0497 1
0498 1 SIDE EFFECTS:
0499 1
0500 1     NONE
0501 1
0502 1 --
0503 1
0504 2 BEGIN
0505 2
0506 2 LOCAL
0507 2     STATUS;
0508 2
0509 2 **
0510 2 !! IF GROUP TEST FAILED IN DATA DEPENDENT MANNER
0511 2 !! READ THE BLOCK BEFORE OVER WRITING IT
0512 2
0513 2     IF
0514 2         .READ_FAIL
0515 2     THEN
0516 2         INCR TEST_INDEX FROM 1 TO TRIALS_TO_SUC DO
0517 2             IF
0518 2                 NOT (STATUS=DO_QIOW(10$_READVBLK+10$_INHRETRY,DISK_TEXT,512,.VBN,0,0))
0519 2             THEN
0520 2                 RETURN .STATUS;
0521 2
0522 2 **
0523 2 !! BLOCK MUST PASS READ/WRITE TEST MULTIPLE BEFORE BEING MARKED GOOD
```



```
531 0524 2
532 0525 INCR TEST_INDEX FROM 1 TO TRIALS_TO_SUC DO
533 0526 BEGIN
534 0527 !*
535 0528 !WRITE TO THE INDICATED DISK BLOCK
536 0529
537 0530 IF
538 0531 NOT (STATUS=DO_QIOW (IOS_WRITEVBLK+IOSM_INHRETRY, GROUP_TEST_DATA, 512, .VBN, 0, 0))
539 0532 THEN
540 0533 RETURN .STATUS;
541 0534
542 0535 !*
543 0536 !TRY AND READ IT BACK
544 0537
545 0538 IF
546 0539 NOT (STATUS=DO_QIOW (IOS_READVBLK+IOSM_INHRETRY, DISK_TEXT, 512, .VBN, 0, 0))
547 0540 THEN
548 0541 RETURN .STATUS;
549 0542
550 0543 !*
551 0544 !MAKE SURE ITS THE SAME
552 0545
553 0546 IF
554 0547 CH$NEQ (512, GROUP_TEST_DATA, 512, DISK_TEXT)
555 0548 THEN
556 0549 RETURN SS$_PARITY
557 0550
558 0551 END;
559 0552
560 0553 RETURN TRUE
561 0554
562 0555 1 END;
```

00FC 0000 BLOCKTEST:

```
57 0000V CF 9E 00002 .WORD Save R2,R3,R4,R5,R6,R7
56 0000 CF 9E 00007 MOVAB DO_QIOW, R7
21 3C00 C6 E9 0000C MOVAB DISK_TEXT, R6
52 01 D0 00011 BLBC READ_FAIL, 2$
7E 04 7E 7C 00014 1$: MOVL #1, TEST_INDEX
04 AC DD 00016 CLRQ -(SP)
7E 0200 8F 3C 00019 PUSHL VBN
7E 8031 56 DD 0001E MOVZWL #512, -(SP)
67 06 FB 00025 PUSHL R6
55 50 D0 00028 MOVZWL #32817, -(SP)
3D 55 E9 0002B CALLS #6, DO_QIOW
52 03 F3 0002E MOVL R0, STATUS
54 01 D0 00032 2$: BLBC STATUS, 4$
7E 7C 00035 3$: MOVL #3, TEST_INDEX, 1$
04 AC DD 00037 MOVL #1, TEST_INDEX
7E 0200 8F 3C 0003A CLRQ -(SP)
1E00 C6 9F 0003F PUSHL VBN
MOVZWL #512, -(SP)
PUSHAB GROUP_TEST_DATA
```

E2

0467

0514

0518

0517

0531

		7E	8030	8F	3C	00043	MOVZWL	#32816, -(SP)	
		67		06	FB	00048	CALLS	#6, DO Q10W	
		55		50	D0	0004B	MOVL	R0, STATUS	
		1A		55	E9	0004E	BLBC	STATUS, 4\$	
				7E	7C	00051	CLRQ	-(SP)	0539
			04	AC	DD	00053	PUSHL	VBN	
		7E	0200	8F	3C	00056	MOVZWL	#512, -(SP)	
				56	DD	0005B	PUSHL	R6	
		7E	8031	8F	3C	0005D	MOVZWL	#32817, -(SP)	
		67		06	FB	00062	CALLS	#6, DO Q10W	
		55		50	D0	00065	MOVL	R0, STATUS	
		04		55	E8	00068	BLBS	STATUS, 5\$	
		50		55	D0	0006B	MOVL	STATUS, R0	0541
				04	0006E		RET		
66	1E00	C6	0200	8F	29	0006F	CMPC3	#512, GROUP_TEST_DATA, DISK_TEXT	0548
				06	13	00077	BEQL	6\$	
		50	01F4	8F	3C	00079	MOVZWL	#500, R0	0550
					04	0007E	RET		
B2		54		03	F3	0007F	AOBLEQ	#3, TEST_INDEX, 3\$	0547
		50		01	D0	00083	MOVL	#1, R0	0553
				04	00086		RET		0555

; Routine Size: 135 bytes, Routine Base: \$CODE\$ + 0128

```
564 0556 1 ROUTINE GROUP_BLOCKTEST=
565 0557 1
566 0558 1 !++
567 0559 1 FUNCTIONAL DESCRIPTION:
568 0560 1
569 0561 1 ROUTINE TESTS GROUPS OF VIRTUALLY CONTIGUOUS BLOCKS FOR
570 0562 1 'BADNESS'. SHOULD ANY OF THE IO OPERATIONS FAIL
571 0563 1 THE STATUS IS IMMEDIATELY RETURNED. GROUPS ARE READ
572 0564 1 SEVERAL TIMES FOR ERROR.
573 0565 1 A WORST CASE IS WRITTEN TO THE GROUP AND THEN READ BACK.
574 0566 1 THE READ DATA IS COMPARED WITH THAT WRITTEN
575 0567 1
576 0568 1 FORMAL PARAMETERS:
577 0569 1
578 0570 1 NONE
579 0571 1
580 0572 1 IMPLICIT INPUTS:
581 0573 1
582 0574 1 STARTING_BLOCK: FIRST VIRTUAL BLOCK IN GROUP
583 0575 1 LAST_BLOCK: LAST VIRTUAL BLOCK IN GROUP
584 0576 1
585 0577 1 IMPLICIT OUTPUTS:
586 0578 1
587 0579 1 NONE
588 0580 1
589 0581 1 ROUTINE VALUE:
590 0582 1 COMPLETION CODES:
591 0583 1
592 0584 1 NONE
593 0585 1
594 0586 1 SIDE EFFECTS:
595 0587 1
596 0588 1 NONE
597 0589 1
598 0590 1 --
599 0591 1
600 0592 2 BEGIN
601 0593 2
602 0594 2 LOCAL
603 0595 2 CURRENT_SIZE,
604 0596 2 STATUS;
605 0597 2
606 0598 2
607 0599 2 !*
608 0600 2 !FOR SHORT FILES OR FOR THE START OF A FILE, GROUP SIZE MAY BE SHORTER
609 0601 2 !THAN THE DEFAULT
610 0602 2
611 0603 2 IF
612 0604 2 .STARTING_BLOCK EQL 1
613 0605 2 THEN
614 0606 2 CURRENT_SIZE=.LAST_BLOCK*512
615 0607 2 ELSE
616 0608 2 CURRENT_SIZE=GROUP_SIZE;
617 0609 2 !*
618 0610 2 !DEFAULT THAT FAILURES WILL NOT BE DATA SENSITIVE
619 0611 2
620 0612 2 READ_FAIL=FALSE;
```



```

621 0613
622 0614
623 0615
624 0616
625 0617
626 0618
627 0619
628 0620
629 0621
630 0622
631 0623
632 0624
633 0625
634 0626
635 0627
636 0628
637 0629
638 0630
639 0631
640 0632
641 0633
642 0634
643 0635
644 0636
645 0637
646 0638
647 0639
648 0640
649 0641
650 0642
651 0643
652 0644
653 0645
654 0646
655 0647
656 0648
657 0649
658 0650
659 0651
660 0652
661 0653
662 0654
663 0655
664 0656
665 0657
666 0658
667 0659
668 0660

! *
! GROUP FAILURE MAY BE DATA SENSITIVE
! READ SEVERAL TIMES BEFORE PASSING TO WRITE/READ TESTING
    INCR TEST_INDEX FROM 1 TO TRIALS_TO_SUC DO
        IF
            NOT (STATUS=DO_QIOW(10$_READVBLK+10$M_INHRETRY,DISK_TEXT,.CURRENT_SIZE,.STARTING_BLOCK,0,0))
        THEN
            BEGIN
                READ FAIL=TRUE;
                RETURN .STATUS
            END;
! *
! GROUP MUST PASS WRITE/READ TEST MULTIPLE TIMES BEFORE
! BEING CONSIDERED GOOD
    INCR TEST_INDEX FROM 1 TO TRIALS_TO_SUC DO
        BEGIN
            ! *
            ! WRITE TO THE INDICATED DISK BLOCK
            IF
                NOT(STATUS=DO_QIOW(10$_WRITEVBLK+10$M_INHRETRY,GROUP_TEST_DATA,.CURRENT_SIZE,.STARTING_BLOCK
            THEN
                RETURN .STATUS;
            ! *
            ! TRY AND READ IT BACK
            IF
                NOT(STATUS=DO_QIOW(10$_READVBLK+10$M_INHRETRY,DISK_TEXT,.CURRENT_SIZE,.STARTING_BLOCK,0,0))
            THEN
                RETURN .STATUS;
            ! *
            ! MAKE SURE ITS THE SAME
            IF
                CH$NEQ(.CURRENT_SIZE,GROUP_TEST_DATA,.CURRENT_SIZE,DISK_TEXT)
            THEN
                RETURN SS$_PARITY
            END;
        RETURN TRUE
    END;
END;

```

01FC 00000 GROUP_BLOCKTEST:

58	0000V	CF	9E	00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8
57	0000'	CF	9E	00007	MOVAB	DO QIOW, R8
					MOVAB	STARTING_BLOCK, R7

: 0556
:
:

		01		67	D1	0000C	CMPL	STARTING_BLOCK, #1	0604
				07	12	0000F	BNEQ	1\$	
54	04	A7		09	78	00011	ASHL	#9, LAST_BLOCK, CURRENT_SIZE	0606
				05	11	00016	BRB	2\$	
		54	1E00	8F	3C	00018	MOVZWL	#7680, CURRENT_SIZE	0608
			F4	A7	D4	0001D	CLRL	READ_FAIL	0612
		52		01	D0	00020	MOVL	#1, TEST_INDEX	0620
				7E	7C	00023	CLRQ	-(SP)	
				67	DD	00025	PUSHL	STARTING_BLOCK	
				54	DD	00027	PUSHL	CURRENT_SIZE	
			C3F4	C7	9F	00029	PUSHAB	DISK_TEXT	
			8031	8F	3C	0002D	MOVZWL	#32817, -(SP)	
		7E		06	FB	00032	CALLS	#6, DO_QIOW	
		68		50	D0	00035	MOVL	R0, STATUS	
		56		56	E8	00038	BLBS	STATUS, 4\$	
		06		01	D0	0003B	MOVL	#1, READ_FAIL	0623
	F4	A7		37	11	0003F	BRB	6\$	0624
DE		52		03	F3	00041	AOBLEQ	#3, TEST_INDEX, 3\$	0619
		55		01	D0	00045	MOVL	#1, TEST_INDEX	0630
				7E	7C	00048	CLRQ	-(SP)	0636
				67	DD	0004A	PUSHL	STARTING_BLOCK	
				54	DD	0004C	PUSHL	CURRENT_SIZE	
			E1F4	C7	9F	0004E	PUSHAB	GROUP_TEST_DATA	
			8030	8F	3C	00052	MOVZWL	#32816, -(SP)	
		7E		06	FB	00057	CALLS	#6, DO_QIOW	
		68		50	D0	0005A	MOVL	R0, STATUS	
		56		56	E9	0005D	BLBC	STATUS, 6\$	
		18		7E	7C	00060	CLRQ	-(SP)	0644
				67	DD	00062	PUSHL	STARTING_BLOCK	
				54	DD	00064	PUSHL	CURRENT_SIZE	
			C3F4	C7	9F	00066	PUSHAB	DISK_TEXT	
			8031	8F	3C	0006A	MOVZWL	#32817, -(SP)	
		7E		06	FB	0006F	CALLS	#6, DO_QIOW	
		68		50	D0	00072	MOVL	R0, STATUS	
		56		56	E8	00075	BLBS	STATUS, 7\$	
		04		56	D0	00078	MOVL	STATUS, R0	0646
		50			04	0007B	RET		
C3F4	C7	E1F4	C7	54	29	0007C	CMPC3	CURRENT_SIZE, GROUP_TEST_DATA, DISK_TEXT	0653
				06	13	00084	BEQL	8\$	
				50	8F	00086	MOVZWL	#500, R0	0655
					04	0008B	RET		
B8		55		03	F3	0008C	AOBLEQ	#3, TEST_INDEX, 5\$	0652
		50		01	D0	00090	MOVL	#1, R0	0658
				04	00093	RET			0660

; Routine Size: 148 bytes, Routine Base: \$CODE\$ + 01AF

```

670 0661 1 ROUTINE DO_QIOW(FUNCTION,P1,P2,P3,P4,P5)=
671 0662 1
672 0663 1 **
673 0664 1 FUNCTIONAL DESCRIPTION:
674 0665 1
675 0666 1 COMMON ROUTINE FOR PERFORMING $QIOW SYSTEM SERVICE
676 0667 1
677 0668 1 FORMAL PARAMETERS:
678 0669 1
679 0670 1 FUNCTION:THE QIOW FUNCTION CODE
680 0671 1 P1: THE ADDRESS OF THE P1 PARAMETER
681 0672 1 P2: THE ADDRESS OF THE P2 PARAMETER
682 0673 1 P3: THE ADDRESS OF THE P3 PARAMETER
683 0674 1 P4: THE ADDRESS OF THE P4 PARAMETER
684 0675 1
685 0676 1 IMPLICIT INPUTS:
686 0677 1
687 0678 1 CHANNEL: THE CHANNEL NUMBER TO THE FILES ACP
688 0679 1 IOSB: THE IO STATUS BLOCK
689 0680 1
690 0681 1 IMPLICIT OUTPUTS:
691 0682 1
692 0683 1 NONE
693 0684 1
694 0685 1 ROUTINE VALUE:
695 0686 1 COMPLETION CODES:
696 0687 1
697 0688 1 RETURNS THE SYSTEM SERVICE CODE FOR THE $QIOW
698 0689 1
699 0690 1
700 0691 1 SIDE EFFECTS:
701 0692 1
702 0693 1 NONE
703 0694 1
704 0695 1 --
705 0696 1
706 0697 2 BEGIN
707 0698 2 LOCAL
708 0699 2 STATUS;
709 0700 2
710 0701 2 !DO QIOW AND CHECK IO SERVICE RETURN
711 0702 2
712 0703 2 IF
713 0704 2 NOT (STATUS=$QIOW(CHAN=.CHANNEL,IOSB=IOSB,
714 0705 2 FUNC=.FUNCTION,
715 0706 2 P1=.P1,
716 0707 2 P2=.P2,
717 0708 2 P3=.P3,
718 0709 2 P4=.P4,
719 0710 2 P5=.P5))
720 0711 2 THEN
721 0712 2 RETURN .STATUS;
722 0713 2
723 0714 2
724 0715 2 !CHECK IO COMPLETION RETURN
725 0716 2
726 0717 2 IF
```

SCANFILE
V04-000

M 16
15-Sep-1984 23:36:57
14-Sep-1984 11:54:33

VAX-11 Bliss-32 V4.0-742
[BADBLK.SRC]SCANFILE.B32;1

Page 22
(10)

```
: 727      0718 2      NOT .IOSB[0]
: 728      0719 2      THEN
: 729      0720 2      RETURN .IOSB[0]
: 730      0721 2      ELSE
: 731      0722 2      RETURN TRUE
: 732      0723 2
: 733      0724 1 END;
```

```
                                .EXTRN  SYSSQIOW
                                DO_QIOW: .WORD  Save R2
52      0000'  CF  9E 00002  MOVAB  IOSB, R2
7E      14    AC  7D 00007  CLRL   -(SP)
7E      0C    AC  7D 0000D  MOVQ   P4, -(SP)
       08    AC  DD 00011  MOVQ   P2, -(SP)
       7E    7C 00014  PUSHL  P1
       52    DD 00016  CLRL   -(SP)
       04    AC  DD 00018  PUSHL  R2
7E      0000G CF  3C 0001B  PUSHL  FUNCTION
       7E    D4 00020  MOVZWL  CHANNEL, -(SP)
00000000G 00    0C  FB 00022  CLRL   -(SP)
       0A    50  E9 00029  CALLS  #12, SYSSQIOW
       04    62  E8 0002C  BLBC   STATUS, 2$
       50    62  3C 0002F  BLBS   IOSB, 1$
       04    04 00032  MOVZWL  IOSB, R0
       50    01  D0 00033  1$:  RET
       04    04 00036  2$:  RET
```

; Routine Size: 55 bytes, Routine Base: \$CODE\$ + 0243

```
: 0661
: 0710
:
:
:
: 0718
: 0722
: 0724
```



```
735 0725 1 GLOBAL ROUTINE DATA_INIT:NOVALUE=
736 0726 1
737 0727 1 ++
738 0728 1 FUNCTIONAL DESCRIPTION:
739 0729 1
740 0730 1     INITIALIZES TEST BLOCKS WITH THE WORST CASE PATTERN
741 0731 1
742 0732 1 FORMAL PARAMETERS:
743 0733 1
744 0734 1     NONE
745 0735 1
746 0736 1 IMPLICIT INPUTS:
747 0737 1
748 0738 1     GROUP_TEST_DATA: BUFFER USED TO WRITE GROUPS OF BLOCKS
749 0739 1
750 0740 1 IMPLICIT OUTPUTS:
751 0741 1
752 0742 1     NONE
753 0743 1
754 0744 1 ROUTINE VALUE:
755 0745 1 COMPLETION CODES:
756 0746 1
757 0747 1     NONE
758 0748 1
759 0749 1 SIDE EFFECTS:
760 0750 1
761 0751 1     NONE
762 0752 1
763 0753 1 --
764 0754 1
765 0755 2 BEGIN
766 0756 2 REGISTER
767 0757 2     POINTER,
768 0758 2     END_POINTER;
769 0759 2 LITERAL
770 0760 2     WORST_CASE_PAT=X0'165555'^16+X0'133333';
771 0761 2
772 0762 2 ++
773 0763 2 INIT POINTERS TO BUFFER
774 0764 2
775 0765 2     POINTER=GROUP_TEST_DATA[0];
776 0766 2     END_POINTER=(GROUP_SIZE^4)+.POINTER;
777 0767 2
778 0768 2 ++
779 0769 2 FILL BUFFER WITH WORST CASE PATTERN
780 0770 2
781 0771 2     WHILE .POINTER NEQU .END_POINTER DO
782 0772 2         BEGIN
783 0773 2             .POINTER=WORST_CASE_PAT;
784 0774 2             POINTER=.POINTER+4
785 0775 2         END;
786 0776 2
787 0777 2     RETURN
788 0778 2
789 0779 2
790 0780 1     END;
```

		0000	00000		.ENTRY	DATA INIT, Save nothing	: 0725
50	0000'	CF	9E 00002		MOVAB	GROUP TEST DATA, POINTER	: 0765
51	01E0	C0	9E 00007		MOVAB	480(R0), END_POINTER	: 0766
51		50	D1 0000C	1\$:	CMPL	POINTER, END_POINTER	: 0771
		09	13 0000F		BEQL	2\$: 0773
80	EB6DB6DB	8F	D0 00011		MOVL	#-345131301, (POINTER)+	: 0774
		F2	11 00018		BRB	1\$: 0780
			04 0001A	2\$:	RET		

: Routine Size: 27 bytes, Routine Base: \$CODE\$ + 027A

```
792 0781 1 ROUTINE CHECK_BADSTATUS(STATUS)=
793 0782 1
794 0783 1 ++
795 0784 1 FUNCTIONAL DESCRIPTION:
796 0785 1
797 0786 1 ROUTINE CLASSIFYS THE SYSTEM SERVICE CODES THAT IT RECEIVES
798 0787 1 AS INPUT INTO 3 CATAGORIES
799 0788 1 NORMAL_STS: $$$ NORMAL
800 0789 1 BAD_STS: DEVICE ERROR INDICATING A BAD BLOCK
801 0790 1 ERROR_STS: UNRECOVERABLE DEVICE ERROR
802 0791 1
803 0792 1 FORMAL PARAMETERS:
804 0793 1
805 0794 1 STATUS: A SYSTEM SERVICE CODE
806 0795 1
807 0796 1 IMPLICIT INPUTS:
808 0797 1
809 0798 1 NONE
810 0799 1
811 0800 1 IMPLICIT OUTPUTS:
812 0801 1
813 0802 1 NONE
814 0803 1
815 0804 1 ROUTINE VALUE:
816 0805 1 COMPLETION CODES:
817 0806 1 RETURNS AS A VALUE ON OF THE 3 ABOVE MENTIONED CODES
818 0807 1 NORMAL_STS,ERROR_STS,BAD_STS
819 0808 1 NONE
820 0809 1
821 0810 1 SIDE EFFECTS:
822 0811 1
823 0812 1 NONE
824 0813 1
825 0814 1 --
826 0815 1
827 0816 2 BEGIN
828 0817 2
829 0818 2 !*
830 0819 2 !POSSIBLE IO CODES ARE DIVIDED INTO THREE CASES
831 0820 2 !GOOD BLOCKS,BAD BLOCKS AND SEVERE DEVICE ERRORS
832 0821 2
833 0822 2 SELECTONE .STATUS OF
834 0823 2 SET
835 0824 2 [$$$_NORMAL]:RETURN NORMAL_STS;
836 0825 2 [$$$_PARITY,
837 0826 2 $$$_CTRLERR,
838 0827 2 $$$_DRVERR]: RETURN BAD_STS;
839 0828 2 [OTHERWISE]:RETURN ERROR_STS
840 0829 2 TES:
841 0830 1 END;
```

0000 00000 CHECK_BADSTATUS:
.WORD Save nothing

: 0781

SCANFILE
V04-000

E 1
15-Sep-1984 23:36:57
14-Sep-1984 11:54:33

VAX-11 Bliss-32 V4.0-742
[BADBLK.SRC]SCANFILE.B32;1

Page 26
(12)

	50	04	AC	D0	00002	MOVL	STATUS, R0	:	0822
	01		50	D1	00006	CMPL	R0, #1	:	0824
			03	12	00009	BNEQ	1\$:	
			50	D4	0000B	CLRL	R0	:	
				04	0000D	RET		:	
00000054	BF		50	D1	0000E	CMPL	R0, #84	:	0825
			12	13	00015	BEQL	2\$:	
0000008C	BF		50	D1	00017	CMPL	R0, #140	:	
			09	13	0001E	BEQL	2\$:	
000001F4	8F		50	D1	00020	CMPL	R0, #500	:	
			04	12	00027	BNEQ	3\$:	
	50		02	D0	00029	MOVL	#2, R0	:	0827
				04	0002C	RET		:	
	50		01	D0	0002D	MOVL	#1, R0	:	0828
				04	00030	RET		:	0830

; Routine Size: 49 bytes, Routine Base: \$CODE\$ + 0295


```

843 0831 1 ROUTINE NORMAL_COMPLETE:NOVALUE=
844 0832 1
845 0833 1 !++
846 0834 1 FUNCTIONAL DESCRIPTION:
847 0835 1
848 0836 1 CALLED AFTER ENTIRE FILE HAS BEEN SCANNED FOR BAD BLOCKS
849 0837 1 ANY OF THE FILE REMAINING IS GOOD AND SHOULD BE
850 0838 1 RETURNED TO THE VOLUME. FILE IS DELETED AND DEACCESSED
851 0839 1
852 0840 1 FORMAL PARAMETERS:
853 0841 1
854 0842 1 NONE
855 0843 1
856 0844 1 IMPLICIT INPUTS:
857 0845 1
858 0846 1 FIB: FILE IDENTIFICATION OF CURRENT FILE
859 0847 1
860 0848 1 IMPLICIT OUTPUTS:
861 0849 1
862 0850 1 NONE
863 0851 1
864 0852 1 ROUTINE VALUE:
865 0853 1 COMPLETION CODES:
866 0854 1
867 0855 1 NONE
868 0856 1
869 0857 1 SIDE EFFECTS:
870 0858 1
871 0859 1 NONE
872 0860 1
873 0861 1 !--
874 0862 1
875 0863 2 BEGIN
876 0864 2 LOCAL
877 0865 2 STATUS:
878 0866 2
879 0867 2 !*
880 0868 2 !TRUNCATE ANY OF THE FILE THAT REMAINS
881 0869 2
882 0870 2 STATUS=TRUNCATE(1);
883 0871 2 IF
884 0872 2 (.STATUS NEQ SS$_NORMAL) AND
885 0873 2 (.STATUS NEQ SS$_ENDOFFILE)
886 0874 2 THEN
887 0875 2 BEGIN
888 0876 2 ERROR_COMPLETE();
889 0877 2 RETURN
890 0878 2 END;
891 0879 2
892 0880 2 !*
893 0881 2 !DELETE THE FILE
894 0882 2
895 0883 2 IF
896 0884 2 NOT DO_OIOW(IO$_DELETE+IOSM_DELETE,FIB_DESC,0,0,0,0)
897 0885 2 THEN
898 0886 2 BEGIN
899 0887 2 ERROR_COMPLETE();

```

```
.. 900      0888      RETURN
.. 901      0889      END;
.. 902      0890
.. 903      0891      !*
.. 904      0892      !DEACCESS THE FILE
.. 905      0893
.. 906      0894      DO_QIOW(10$,DEACCESS,FIB_DESC,0,0,0,0);
.. 907      0895
.. 908      0896      RETURN
.. 909      0897      END;
```

```
0000 00000 NORMAL_COMPLETE:
                                .WORD      Save nothing
                                PUSHL      #1
                                CALLS      #1, TRUNCATE
                                CMPL       STATUS, #1
                                BEQL       1$
                                CMPL       STATUS, #2160
                                BNEQ       2$
                                CLRQ      -(SP)
                                CLRQ      -(SP)
                                PUSHAB     FIB_DESC
                                MOVZWL     #309, -(SP)
                                CALLS      #6, DO_QIOW
                                BLBS       R0, 3$
                                CALLS      #0, ERROR_COMPLETE
                                RET
                                CLRQ      -(SP)
                                CLRQ      -(SP)
                                PUSHAB     FIB_DESC
                                PUSHL      #52
                                CALLS      #6, DO_QIOW
                                RET
```

FDDA	CF	01	01 DD 00002
			01 FB 00004
			50 D1 00009
			09 13 0000C
00000870	8F		50 D1 0000E
			15 12 00015
			7E 7C 00017 1\$:
			7E 7C 00019
		0000'	CF 9F 0001B
		0135	8F 3C 0001F
FF54	7E		06 FB 00024
	CF		50 E8 00029
0000V	06		00 FB 0002C 2\$:
	CF		04 00031
			7E 7C 00032 3\$:
			7E 7C 00034
		0000'	CF 9F 00036
FF3C	CF		34 DD 0003A
			06 FB 0003C
			04 00041

; Routine Size: 66 bytes, Routine Base: \$CODE\$ + 02C6

```
.. 0831
.. 0870
.. 0872
.. 0873
.. 0884
..
..
.. 0887
.. 0886
.. 0894
..
.. 0897
```

```

911 0898 1 ROUTINE ERROR_COMPLETE:NOVALUE=
912 0899 1
913 0900 1 ++
914 0901 1 FUNCTIONAL DESCRIPTION:
915 0902 1
916 0903 1 CALLED WHEN A FATAL DEVICE ERROR OR SYSTEM SERVICE ERROR
917 0904 1 IS ENCOUNTERED DURING PROCESSING. THE CURRENT FILE IS DEACCESSED
918 0905 1
919 0906 1 FORMAL PARAMETERS:
920 0907 1
921 0908 1 NONE
922 0909 1
923 0910 1 IMPLICIT INPUTS:
924 0911 1
925 0912 1 NONE
926 0913 1
927 0914 1 IMPLICIT OUTPUTS:
928 0915 1
929 0916 1 NONE
930 0917 1
931 0918 1 ROUTINE VALUE:
932 0919 1 COMPLETION CODES:
933 0920 1
934 0921 1 NONE
935 0922 1
936 0923 1 SIDE EFFECTS:
937 0924 1
938 0925 1 NONE
939 0926 1
940 0927 1 --
941 0928 1
942 0929 2 BEGIN
943 0930 2
944 0931 2 !*
945 0932 2 !DEACCESS THE FILE
946 0933 2
947 0934 2 DO_QIOW(10$_DEACCESS,FIB_DESC,0,0,0,0);
948 0935 2
949 0936 2
950 0937 2 RETURN
951 0938 1 END;

```

0000 00000 ERROR_COMPLETE:

				WORD	Save nothing
	7E	7C	00002	CLRD	-(SP)
	7E	7C	00004	CLRD	-(SP)
0000*	CF	9F	00006	PUSHAB	FIB_DESC
	34	DD	0000A	PUSHL	#52
FF2A CF	06	FB	0000C	CALLS	#6, DO_QIOW
	04	00	00011	RET	

; Routine Size: 18 bytes, Routine Base: \$CODE\$ + 0308

0898
0934
0938

SCANFILE
V04-000

1
15-Sep-1984 23:36:57
14-Sep-1984 11:54:33

VAX-11 Bliss-32 V4.0-742
[BADBLK.SRC]SCANFILE.B32;1

Page 30
(14)


```

953 0939 1 ROUTINE GROUP_RETURN=
954 0940 1 **
955 0941 1 FUNCTIONAL DESCRIPTION:
956 0942 1
957 0943 1
958 0944 1 CALLED WHEN A BAD BLOCK ERROR IS ENCOUNTERED BY
959 0945 1 GROUP BLOCK TESTING. THE INDIVIDUAL BLOCKS IN A GROUP
960 0946 1 ARE TESTED FOR 'BADNESS' AND TRUNCATED OFF THE CURRENT
961 0947 1 FILE AND INTO THE BAD BLOCK FILE WHEN FOUND
962 0948 1 FORMAL PARAMETERS:
963 0949 1
964 0950 1 NONE
965 0951 1
966 0952 1 IMPLICIT INPUTS:
967 0953 1
968 0954 1 STARTING_BLOCK: FIRST BLOCK IN GROUP
969 0955 1 LAST_BLOCK: LAST BLOCK IN GROUP
970 0956 1
971 0957 1 IMPLICIT OUTPUTS:
972 0958 1
973 0959 1 NONE
974 0960 1
975 0961 1 ROUTINE VALUE:
976 0962 1 COMPLETION CODES:
977 0963 1
978 0964 1 NONE
979 0965 1
980 0966 1 SIDE EFFECTS:
981 0967 1
982 0968 1 NONE
983 0969 1
984 0970 1 --
985 0971 1
986 0972 2 BEGIN
987 0973 2 LOCAL
988 0974 2 VBN;
989 0975 2
990 0976 2 !*
991 0977 2 !INDIVIDUALLY CONSIDER ALL BLOCKS IN THE GROUP
992 0978 2 !RETURN EACH TO THE BADBLOCK FILE OR FREE SPACE
993 0979 2
994 0980 2 VBN=.LAST_BLOCK;
995 0981 2 WHILE TRUE DO
996 0982 2 BEGIN
997 0983 2
998 0984 2 CASE CHECK_BADSTATUS(BLOCKTEST(.VBN))
999 0985 2 FROM NORMAL_STS TO BAD_STS OF
1000 0986 2 SET
1001 0987 2
1002 0988 2 [NORMAL_STS]:TRUNC_BLOCK=.VBN;
1003 0989 2
1004 0990 2 [ERROR_STS]:RETURN FALSE;
1005 0991 2
1006 0992 2 [BAD_STS]:TRUNCATE_BAD(.VBN);
1007 0993 2
1008 0994 2 TES;
1009 0995 2 VBN=.TRUNC_BLOCK-1;

```

```

: 1010      0996      3      IF
: 1011      0997      3      .VBN LSS .STARTING_BLOCK
: 1012      0998      3      THEN
: 1013      0999      3      RETURN TRUE
: 1014      1000      3      END
: 1015      1001      3
: 1016      1002      1 END;

```

```

                                000C 00000 GROUP_RETURN:
                                .WORD
                                Save R2,R3
                                TRUNC_BLOCK, R3
                                LAST_BLOCK, VBN
                                VBN
                                #1, BLOCKTEST
                                R0
                                #1, CHECK_BADSTATUS
                                R0, #0, #2
                                3$-2$,-
                                4$-2$,-
                                5$-2$
                                VBN, TRUNC_BLOCK
                                6$
                                R0
                                VBN
                                #1, TRUNCATE_BAD
                                #1, TRUNC_BLOCK, VBN
                                VBN, STARTING_BLOCK
                                1$
                                #1, R0
                                0000'  CF  9E 00002  MOVAB
                                OC   A3  D0 00007  MOVL
                                52   DD 0000B 1$:  PUSHL
                                FDFC  CF  01  FB 0000D  CALLS
                                50   DD 00012  PUSHL
                                FF62  CF  01  FB 00014  CALLS
                                02    00    50  CF 00019  CASEL
                                000E  000B  0006  0001D 2$:  .WORD
                                63          52  D0 00023 3$:  MOVL
                                OA   11 00026  BRB
                                50   D4 00028 4$:  CLRL
                                04 0002A  RET
                                52  DD 0002B 5$:  PUSHL
                                FD8B  CF  01  FB 0002D  CALLS
                                52    63  01  C3 00032 6$:  SUBL3
                                08    A3  52  D1 00036  CMPL
                                CF   18 0003A  BGEQ
                                50    01  D0 0003C  MOVL
                                04 0003F  RET

```

```

: 0939
: 0980
: 0984
:
:
:
:
: 0988
: 0990
: 0992
: 0995
: 0997
: 0999
: 1002

```

; Routine Size: 64 bytes, Routine Base: \$CODE\$ + 031A

SCANFILE
V04-000

1
15-Sep-1984 23:36:57
14-Sep-1984 11:54:33

VAX-11 Bliss-32 V4.0-742
[BADBLK.SRC]SCANFILE.B32;1

Page 33
(18)

: 1020 1003 1 END
: 1021 1004 0 ELUDOM

!End of module

PSECT SUMMARY

Name	Bytes	Attributes
\$OWNS	15476	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$PLITS	8	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$CODES	858	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA2B:[SYSLIB]LIB.L32;1	18619	27	0	1000	00:01.8

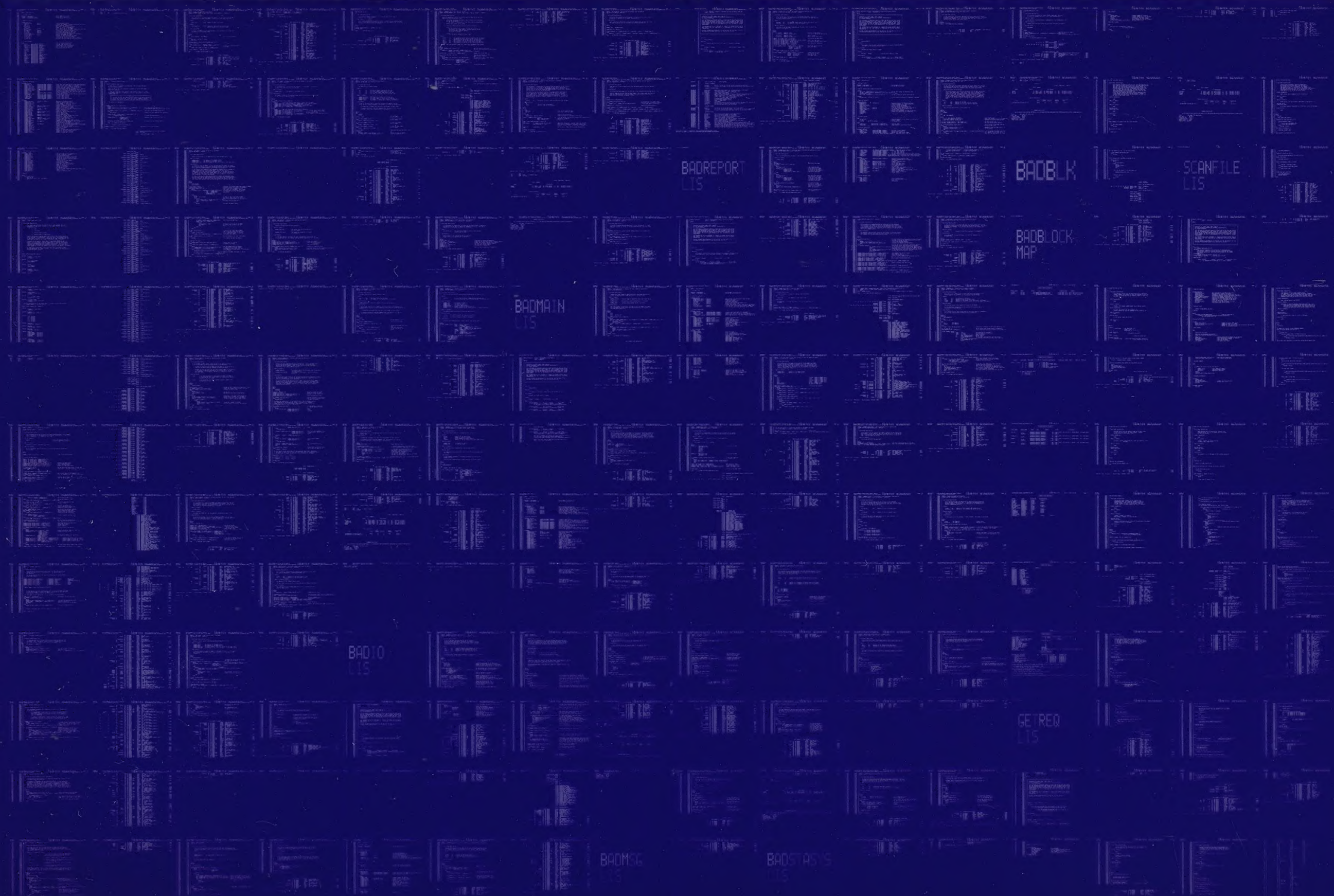
COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:SCANFILE/OBJ=OBJ\$:SCANFILE MSRC\$:SCANFILE/UPDATE=(ENHS:SCANFILE)

: Size: 858 code + 15484 data bytes
: Run Time: 00:17.7
: Elapsed Time: 00:33.5
: Lines/CPU Min: 3407
: Lexemes/CPU-Min: 9369
: Memory Used: 91 pages
: Compilation Complete

0018 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



0019 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY